A. An Example Comparing GeoCert and LayerCert-Basic

The following simple example of a network with two hidden layers over a 2D input, demonstrates the advantage of LayerCert-Basic over GeoCert. Recall the definition of the neural network from (2). We pick the following values for weights and biases of the network:

\[
W_0 := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad b_0 := \begin{bmatrix} 0 \\ 0 \end{bmatrix},
\]
\[
W_1 := \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad b_1 := -1,
\]
\[
W_2 := \begin{bmatrix} 1 \end{bmatrix}, \quad b_2 := \begin{bmatrix} -5 \\ 0 \end{bmatrix}.
\]

Let the input point be \( x := [-1, -1.25]^T \). The input space and the corresponding graphs for LayerCert and GeoCert are illustrated in Figure 7. We leave the decision boundary off the figure.

![LayerCert and GeoCert graphs](image)

*Figure 7. Input space with hierarchical graph for LayerCert and neighborhood graph for GeoCert.*

The next figure shows the order in which GeoCert processes the nodes and the edges (distance computations). At each node, we check if there is a decision boundary contained within the region. As for each edge, we compute the distance from the input \( x \) to the boundary of the two regions that is connected by the edge.

![GeoCert node and edge order](image)

*Figure 8. GeoCert Node and Edge Processing Order.*

For LayerCert, we visit more nodes. Note that we still visit all nodes in the last layer in the same relative order. We only need to check for decision boundaries for the nodes in the last layer. In addition, within the first layer, we do not have to compute the distance to the top-right green node more than once.

![LayerCert node and edge order](image)

*Figure 9. LayerCert Node and Edge Processing Order.*
B. Proofs for LayerCert-Basic

B.1. Exploring Hyperplane Arrangements

Before we prove the result for the full hierarchical structure, we focus on the simpler case of just hyperplane arrangements.

We show that for each pattern $A$, dist$(R_A, x)$ can be obtained by considering any neighboring pattern $A'$ such that $R_{A'}$ is closer to $x$ than $R_A$ is. This fact is allows us to skip the computation of some of the distances compared to GeoCert.

**Proposition B.1.** Let dist be a convex distance function. Given a region $R_A$ and a neighboring region $R_{A'}$ such that dist$(R_A, x) \geq$ dist$(R_{A'}, x)$, we have dist$(R_A, x) = \text{dist(Face}(A, A'), x)$.

Proposition B.1 is a special case of the following lemma:

**Lemma B.2.** Let $C$ be a closed convex set and $f, g : \mathbb{R}^n \to \mathbb{R}$ be convex functions. Suppose $D := C \cap \{v \mid g(v) \leq 0\}$ is nonempty and suppose $\min_{x \in C} f(x) \leq \min_{x \in D} f(x)$. Then there exists $x^* \in \arg\min_{x \in D} f(x)$ satisfying $g(x^*) = 0$.

**Proof.** Pick some $z \in \arg\min_{x \in C} f(v)$ and $y \in \arg\min_{x \in D} f(v)$. If $g(z) = 0$ or $g(y) = 0$, we are done. Suppose then that $g(z) > 0$ and $g(y) < 0$. Let $w = z - y$ and pick $\lambda \in (0, 1)$ such that $g(y + \lambda w) = 0$. It follows that $y + \lambda w$ is in $D$ and by the convexity of $f$, $f(y + \lambda w) \leq f(y)$. Hence $y + \lambda w$ satisfies the claim.

The next thing we show about hyperplane arrangements will be useful when we consider nested hyperplane structures.

**Proposition B.3.** Let $C$ be a convex set in $\mathbb{R}^n$ and consider a hyperplane arrangement. Let dist be a convex distance function. Let $A$ be a pattern that contains a point in $\arg\min_{v \in C} \text{dist}(v, x)$ and consider another pattern $A'$ such that $R_{A'} \cap C$ is nonempty. Then there is a sequence of patterns $A = A_1, A_2, \ldots, A_k, A_{k+1} = A'$ such that we have

- $A_i$ and $A_{i+1}$ are neighboring patterns for $i \in [k]$,
- $\text{dist}(R_{A_i}, C, x) \leq \text{dist}(R_{A_{i+1}}, C, x)$ for $i \in [k]$, and
- $R_{A_i} \cap C$ is nonempty for $i \in [k]$.

Before we prove Proposition B.3, we will prove the following result about the hyperplane arrangements. We use the terms ‘regions’ and ‘neighbors’ here similar to their use in Definitions 3.4 and 3.5.

**Lemma B.4.** Consider a hyperplane arrangement, a convex set $C$, and a line with endpoints contained entirely within $C$. Consider the undirected graph where the nodes are the regions of the arrangement and edges are formed between neighboring regions. For all regions in the arrangement that have a nonempty intersection with the line, there is a path between two regions that only goes through the regions touching the line.

**Proof.** We will first show that any two regions that share a common point can be connected through regions that share that same point $v$. Suppose this is true if there are $k$ hyperplanes going through the point $v$. Let $A$ and $A'$ be neighboring regions that share the point $v$. After introducing another hyperplane through $v$, suppose $A$ gets split into two neighboring regions $A_1, A_2$. The addition of the hyperplane either splits $A'$ into two regions or keeps it as one. Let $A^3$ be one of the nonempty regions. It must be the case that $A^3$ is on the same side of the hyperplane as one of $A_1, A_2$, so $A^3$ is a neighbor to one of them. Hence, all the regions around the point remain connected to each other through each other.

We will now show we can move from one end of the line to the other. The hyperplane arrangement partitions the line into line segments $\{l_1, l_2, \ldots, l_k\}$. There is a region that covers each line segment, so we can connect any two such line segment covering regions through a path. Furthermore, any region touching the line must touch one of the endpoints of the line segment, and hence there is a path between that point and the corresponding line segment covering region. This concludes the proof.

**Proof of Proposition B.3.** Let $v, v'$ be the minimizers of dist$(\cdot, x)$ when restricted to $R_A \cap C$ and $R_{A'} \cap C$ respectively. If dist$(R_A \cap C, x) = \text{dist}(R_{A'} \cap C)$, then the straight line segment connecting $v$ and $v'$ only passes through patterns with the same distance by convexity of the distance function and minimality of $A$.

We now perform induction on the distance dist$(R_{A'} \cap C, x)$. Suppose this is true for all regions of distance less than $d$. Pick the region $A'$ of the next lowest distance $d'$. Again consider the straight line segment connecting $v$ and $v'$. All regions
touching $v'$ have distance at most $d'$. Furthermore, since the number of regions is finite, by the convexity of the line segment there must be some region $B$ that contains $v'$ that also contains other points on the line segment closer to $v$. The convexity of the distance function implies that $\text{dist}(R_B \cap C, x) < d'$. By Lemma B.4 there must be a series of neighboring regions connecting $B$ to $A'$, and as a result $A'$ is connected to a region of at most distance $d$ through regions of at most distance $d'$. This concludes the proof.

**B.2. Correctness Proof**

Using the concepts defined in Definitions 3.7 and 3.8, we can construct the following hierarchical graph over the set of all nonempty regions:

- **Root node (level 0):** an empty ‘activation pattern’ corresponding to the ‘activation region’ $R^n$.
- **Nodes at level $l$:** the nonempty layer-$l$ partial activation region.
- **Edges at level $l$:** between $A$ and $A'$ if they are $l$-layer neighbors (i.e. $A$ and $A'$ differ on a single $l$th layer neuron).
- **Edges between levels $l$ and $l + 1$:** between parent($A$) and $A$ if $A$ is obtained by next$_{layer}(A, v)$ during the GeoCert algorithm.

An example of this graph is illustrated in Figure 3. We term this graph the hierarchical search graph. Note that at each iteration of LayerCert, it explores a node neighboring those that have been explored of closest distance and computes the distance to all the neighboring nodes.

**Lemma B.5.** Let $H$ denote the hierarchical search graph of LayerCert-Basic for some network and input point $x$. There is a path in $H$ from the initial activation pattern $A^x$ to every pattern $A$ such that the distances of the patterns on the path are monotonically increasing.

**Proof.** Suppose the main claim holds for all patterns up to layer $l - 1$. We will now show this for layer $l$. The claim immediately holds for any pattern obtained through the next$_{layer}$ operation. Given an arbitrary $l$-layer region $R_A$ of distance $d$ from $x$, as a consequence of Proposition B.3 there is a path in $H$ between $B$ and $A$ of patterns with monotonically increasing distance. Pick the first unpopped pattern $A'$ on this path that is of distance $d$ from $x$. $A'$ must have a popped pattern next to it, and as a result $A'$ must be on the priority queue.

The proof of correctness is a consequence of Lemma B.5.

**Proof of Theorem 4.1.** Since the distance to the distance boundary contained within a full region is further or equal to the distance of the region, it follows that if we process all the regions in order of distance we will terminate only when we have computed all regions closer than the closest decision boundary.

The first region processed is the region associated the empty activation pattern which has distance 0 from the input point. Now suppose that we have popped all patterns of distance $< d$ from the priority queue and the next closest un-popped region is of distance $d$. We will show that the priority queue contains a region of distance $d$ before the next iteration of the algorithm and that we have correctly computed the distance to this region.

Pick an unpopped activation pattern $A$ of distance $d$ of the shortest path length to the root node in the hierarchical search graph $H$. If $A$ is directly connected to its parent parent($A$) in $H$, then we must have popped and processed parent($A$) in some iteration of the algorithm (since that has a smaller distance to the root node) and as a result $A$ is in the priority queue. If this is not the case, then there is some sibling $B$ that is connected to parent($A$). By Lemma B.2, we know that $\text{dist}(R_B, x) \leq \text{dist}(R_A, x)$. Proposition B.3 tells us that there is a path in $H$ between $B$ and $A$ of patterns with monotonically increasing distance. Pick the first unpopped pattern $A'$ on this path that is of distance $d$ from $x$. $A'$ must have a popped pattern next to it, and as a result $A'$ must be on the priority queue.

**B.3. Main Proof**

**Proof of Theorem C.1.** Note that LayerCert and GeoCert process the same full activation patterns since both algorithms process them by distance.
We will first show that the only nodes LayerCert processes are those that are full activation patterns or ancestors of the full activation patterns processed. Suppose for contradiction that this is not the case. Then, there is some processed pattern \( A \) that is not a full activation pattern where we process no child. However, LayerCert will apply next_layer to \( A \), adding the child node \( B \) with the same priority as \( A \) to the priority queue. By assumption we have \( U \) larger than the distance of \( B \), so we will process \( B \) eventually, contradicting our earlier claim.

For any full activation pattern \( A \), there is a one-to-one mapping between the convex programs processed by GeoCert and the convex programs processed LayerCert for \( A \) and all ancestors of \( A \). This is because we form one program for each neuron in the ReLU network. Consider the \( j \)-th neuron in the \( l \)-th layer. The constraints in the convex program formed by GeoCert for \( A \) and a neighboring full pattern \( A' \) are

\[
A_i(j)z_j^i \geq 0 \text{ for } i \in [L], j \in [n_i], \text{ and } z_{j'}^{i'} = 0
\]

where \( z_j^i \) is defined as in (2). In contrast, LayerCert uses

\[
A_i(j)z_j^i \geq 0 \text{ for } i \in [l], j \in [n_i], \text{ and } z_{j'}^{i'} = 0
\]

which only has a subset of the constraints. Since the only nodes we have to consider for LayerCert are ancestors of full activation patterns, this concludes the proof of the theorem.

\[ \square \]

C. Simplifying Projection Problems via Upper Bounds

To reduce the number of constraints in Problems (9) and (7), Jordan et al. (2019) note that we can quickly decide if a constraint of the form \( a^\top x \leq b \) is necessary by checking if the hyperplane \( \{ x \mid a^\top x = b \} \) overlaps with the ball \( B_{p,U}(x) \). This can be done via performing an \( \ell_p \) projection of \( x \) onto the hyperplane given by \( a^\top x = b \), which can be done in a linear number of elementary operations. We note that the technique of using upper and lower bounds on variables (corresponding to the case where \( p = \infty \) to prune constraints is standard in the linear programming literature (see for example Gondzio (1997)). Each time a tighter upper bound \( U \) is obtained, the ball \( B_{p,U}(x) \) shrinks, allowing us to remove more constraints.

We can apply this technique to GeoCert and any variant of the LayerCert method. Under the additional assumption that GeoCert and LayerCert-Basic process full activation patterns in the same order, LayerCert-Basic maintains its advantage in number and size of convex programs solved.

**Theorem C.1.** (Complexity of LayerCert-Basic with Domain-based Constraint Pruning) Suppose LayerCert-Basic and GeoCert process full activation patterns in the same order. Suppose we formulate the convex problems associated with decision_bound and next_layer using Formulations (6) and (8). We can construct an injective mapping from the set of convex programs solved by LayerCert to the corresponding set in GeoCert such that the constraints in the LayerCert program is a subset of those in the corresponding GeoCert program.

The proof of this is similar to the proof of Theorem C.1. The addition of domain-based pruning does not affect the result since the same constraints are pruned for both methods.

D. LayerCert Framework and Convex Restrictions

We expand on the discussion on the use in Section 5 on how to use incomplete verifiers such as those presented in Weng et al. (2018); Zhang et al. (2018) or the full linear programming relaxation of single ReLUs to improve LayerCert. Once we compute a convex set \( M \) that contains the closest decision boundary, we can subsequently restrict our attention to just the intersection of activation regions with \( M \). The correctness of this follows from the fact that Proposition B.3 accommodates the use of such a set \( M \) and from modifying the hierarchical search graph in Lemma B.5.

Instead of just applying restriction at the initiation iteration, one can repeatedly apply it at every iteration. This version of restriction also takes in an activation pattern and forms the restriction based on this. We describe the full algorithm in Algorithm 4. The use of restriction in every iteration can significantly add significant overhead to the algorithm, so there is a trade-off between (1) the choice of restriction used and (2) the frequency in which restriction is applied, and these choices can depend heavily on the width and depth of the neural networks that we are verifying. We leave a computational study of this to future work.
Algorithm 4 LayerCert with Recursive Convex Restrictions

1: **Input:** $x, y$ (label of $x$), $U$ (upper bound)
2: $d, A, M \leftarrow \text{restriction}(\emptyset, x, ub)$
3: $Q \leftarrow \text{empty priority queue}$
4: $Q.push((d, A, M))$
5: $S \leftarrow \emptyset$
6: **while** $Q \neq \emptyset$ **do**
7: $(d, A, M) \leftarrow Q.pop()$
8: **if** $ub \leq d$ **then**
9: **return** $U$
10: **else**
11: **if** $A$ is a full activation pattern **then**
12: $ub \leftarrow \min(\text{decision bound}(A, x, ub)$
13: **else**
14: **if** contains$_{db}(A, x, ub) = \text{`maybe'}$ **then**
15: $(d', A', M') \leftarrow \text{restriction}(A, x, ub)$
16: $Q.push((d', A', M'))$
17: **for** $A' \in N_{current\_layer}(A) \setminus S$ **do**
18: **if** Face$(A, A') \cap M$ is nonempty **then**
19: $d' \leftarrow \text{priority}(x, \text{Face}(A, A') \cap M)$
20: $Q.push((d', A', M))$
21: $S \leftarrow S \cup \{A\}'$

E. Additional Experiments and Details

E.1. Parameter Choices

**Convex programming software.** We used the following package versions and settings for handling convex programs. All the solvers are accessed through CVXPY’s interface:

- **Modeling language:** CVXPY v1.0.25 (Diamond & Boyd, 2016).
- **Primary solver:** ECOS v2.0.7 (Domahidi et al., 2013) – ‘abstol’: $1 \times 10^{-5}$, ‘reftol’: $1 \times 10^{-4}$, ‘feastol’: $1 \times 10^{-5}$, ‘abstol\_inacc’: $5 \times 10^{-3}$, ‘reftol\_inacc’: $5 \times 10^{-2}$, ‘feastol\_inacc’: $5 \times 10^{-3}$.
- **Secondary solver:** OSQP v0.6.0 (Stellato et al., 2017) – ‘eps\_abs’: 0.001, ‘eps\_rel’: 0.001.
- **Backup solver:** SCS v2.1.1 (O’donoghue et al., 2016) – default settings.

**Use of constraint-reducing techniques.** We used the same domain-based techniques mentioned in Section C for all the methods. Before adding a constraint to a convex program to solve, we check if the constraint overlaps with $B_{p,U}(x)$.

E.2. Additional Results

**$\ell_2$-norm experiments.** In Table 2 we present the averaged results for $\ell_2$-norm. We used an upper bound radius of 3.0. As with the $\ell_\infty$ results presented in Table 1, the LayerCert methods consistently outperform the GeoCert methods in terms of the timing. We note that some of the difference in running times is partially due to the solvers – ECOS and OSQP failed frequently, and whenever SCS is invoked the running time is significantly increased. This issue persisted over a range of hyperparameter settings for ECOS and OSQP. We note that this is rare for the $\ell_\infty$-norm experiments.

As for the number of quadratic programs, LayerCert-CROWN and LayerCert-Both consistently outperform both GeoCert methods, while LayerCert and LayerCert-IA mostly but not always outperforms GeoCert-Lip. GeoCert is consistently the worst performing method in terms of number of QPs and running time.

**Ablation Study for LayerCert-Basic.** In Section 4.2, one of the reasons provided for the reduction in the number of priority computations for LayerCert-Basic over GeoCert was that was LayerCert marked a pattern $A$ as ‘seen’ after the
Hierarchical Verification for Adversarial Robustness

Table 2. Average number of convex programs and running time over 100 inputs for 12 different networks for $\ell_2$-norm distance. The green shaded entries indicate the best performing method for each neural network under each metric. The gray shaded entries indicate the algorithm timed out before an exact solution to Problem (1) could be found or before a radius of 3 is reached for at least one input. Whenever a timeout occurs, we use a time of 1800 seconds in its place, which leads to an underestimate of the true time.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#QPs</td>
<td>Time (s)</td>
<td>#QPs</td>
<td>Time (s)</td>
<td>#QPs</td>
<td>Time (s)</td>
<td>#QPs</td>
<td>Time (s)</td>
<td>#QPs</td>
<td>Time (s)</td>
<td>#QPs</td>
<td>Time (s)</td>
</tr>
<tr>
<td>GeoCert</td>
<td>22.0</td>
<td>1.80</td>
<td>56.3</td>
<td>10.47</td>
<td>96.0</td>
<td>6.21</td>
<td>49.3</td>
<td>25.30</td>
<td>64.9</td>
<td>304.80</td>
<td>24.6</td>
</tr>
<tr>
<td>GeoCert-Lip</td>
<td>19.0</td>
<td>1.70</td>
<td>34.9</td>
<td>9.77</td>
<td>51.0</td>
<td>3.58</td>
<td>41.8</td>
<td>22.00</td>
<td>62.7</td>
<td>242.47</td>
<td>22.6</td>
</tr>
<tr>
<td>LayerCert</td>
<td>15.3</td>
<td>0.76</td>
<td>36.4</td>
<td>1.34</td>
<td>53.7</td>
<td>2.03</td>
<td>27.9</td>
<td>1.43</td>
<td>29.0</td>
<td>49.45</td>
<td>16.0</td>
</tr>
<tr>
<td>LayerCert-IA</td>
<td>15.3</td>
<td>0.79</td>
<td>36.4</td>
<td>1.38</td>
<td>53.7</td>
<td>2.07</td>
<td>24.0</td>
<td>1.12</td>
<td>29.0</td>
<td>49.35</td>
<td>16.0</td>
</tr>
<tr>
<td>LayerCert-CROWN</td>
<td>15.3</td>
<td>0.79</td>
<td>28.9</td>
<td>1.20</td>
<td>41.3</td>
<td>1.58</td>
<td>25.4</td>
<td>1.95</td>
<td>27.8</td>
<td>52.12</td>
<td>14.6</td>
</tr>
<tr>
<td>LayerCert-Both</td>
<td>15.3</td>
<td>0.82</td>
<td>28.9</td>
<td>1.24</td>
<td>41.3</td>
<td>1.61</td>
<td>22.7</td>
<td>1.74</td>
<td>27.8</td>
<td>51.87</td>
<td>14.6</td>
</tr>
</tbody>
</table>

Table 3. The results are presented in Table 3. We see that this leads to a slight overhead over LayerCert, but overall this accounts only for a small fraction of the improvement in performance over GeoCert. This indicates that the majority of the improvement comes from hierarchical structure and the way that it allows us to avoid a large number of priority computations while reducing the complexity of each computation.

Experiments for Larger Networks To compare the performance of LayerCert and GeoCert for large networks, we chose the following fully-connected networks and used $\ell_\infty$-distance.

- 8 x [20],
- 6 x [30],
- 5 x [50],
- 5 x [60].

We chose the 10 instances where none of the methods were able to converge within 1800 seconds and compared the performance of GeoCert with Lipschitz term and LayerCert with both heuristics. Figures 10, 11, 12, 13 contain these results. The orange solid line in each represent LayerCert while the blue dotted line GeoCert.
Hierarchical Verification for Adversarial Robustness

Table 3. Ablation Study: Average number of convex programs and running time over 100 inputs for 12 different networks for $\ell_\infty$-norm distance, focusing on GeoCert, LayerCert-Basic, and a variant of LayerCert that allow repeated priority computations. The green shaded entries indicate the best performing method for each neural network under each metric. The gray shaded entries indicates the algorithm timed out before an exact solution to Problem (1) could be found or before a radius of 0.3 is reached for at least one input. Whenever a timeout occurs, we use a time of 1800 seconds in its place, which leads to an underestimate of the true time.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoCert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerCert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerCert + Repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#LPs</td>
<td>Time (s)</td>
<td>#LPs</td>
<td>Time (s)</td>
<td>#LPs</td>
<td>Time (s)</td>
</tr>
<tr>
<td>GeoCert</td>
<td>867.4</td>
<td>88.92</td>
<td>18.0</td>
<td>1.48</td>
<td>392.7</td>
</tr>
<tr>
<td>LayerCert</td>
<td>568.3</td>
<td>29.95</td>
<td>12.4</td>
<td>0.69</td>
<td>203.5</td>
</tr>
<tr>
<td>LayerCert + Repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#LPs</td>
<td>Time (s)</td>
<td>#LPs</td>
<td>Time (s)</td>
<td>#LPs</td>
<td>Time (s)</td>
</tr>
<tr>
<td>GeoCert</td>
<td>440.4</td>
<td>25.52</td>
<td>12.1</td>
<td>0.68</td>
<td>176.7</td>
</tr>
<tr>
<td>LayerCert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10. Timing for 10 Instances, 8 × [20] network.

Figure 11. Timing for 10 Instances, 6 × [30] network.
Hierarchical Verification for Adversarial Robustness

Figure 12. Timing for 10 Instances, 5 × [50] network.

Figure 13. Timing for 10 Instances, 5 × [60] network.